Claim Amendments

Claims 1-100 (canceled)

101. (currently amended) A method for communicating between first and second applications executing on respective first and second independent partitions of a partitionable computer system, wherein each of said first and second partitions operates under the control of a separate operating system, and wherein said first and second applications are configured to communicate with each other via a physical network using standard network interfaces; said method comprising:

receiving a request made by said first application for establishing a network connection with said second application for sending a message thereto; and

in response to said request, establishing an emulated network connection between said first and said second applications through a shared memory region of the computer system shared by said first and said second partitions;

said establishing being such that said emulated network connection permits said first and second applications to communicate with each other via said emulated network connection using standard network interfaces even though there is no physical network therebetween;

said establishing also being such that the receiving of a message by said second application from said first application via said emulated network connection appears to said second application as having been sent via an external physical network even though there is no external physical network.

102. (previously presented) The method recited in claim 101, wherein said emulated network connection requested by said first application comprises a network socket connection, and wherein the step of establishing a connection through the shared memory region comprises establishing a connection through the shared memory region that emulates a network socket connection.

103. (previously presented) The method recited in claim 101, wherein said establishing communication between said first and second applications includes:

creating a data structure in said shared memory region comprising a plurality of data segments for use in sending a message from said first application to said second application.

104. (previously presented) The method recited in claim 103, wherein sending a message from said first application to said second application includes:

writing, from the first partition on behalf of the first application, a message to one or more of said data segments, as needed, and updating an indication of the data segment containing the most recently written portion of the message;

reading from the second partition on behalf of the second application, a message from said one or more data segments and updating an indication of which data segments have been read from the data structure; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

105. (previously presented) The method of claim 104, wherein said plurality of data segments form a circular buffer, and wherein updating an indication of the data segment containing the most recently written portion of the message comprises incrementing a head index.

106. (previously presented) The method of claim 105, wherein updating an indication of which data segments have been read from the data structure comprises incrementing a tail index.

107. (previously presented) The method of claim 106, further comprising polling, by the receiving partition, the shared memory region to determine if the message has been written to the shared memory region.

108. (previously presented) The method of claim 107, further comprising receiving, by the receiving partition, an interrupt initiated by the sending partition and indicating that the message has been written to the shared memory region.

109. (previously presented) The method of claim 102, wherein the step of establishing an emulated socket connection between the first partition and the second partition further comprises performing the following steps on the second partition:

(a) creating an emulated socket connection on behalf of the second application to listen for attempts to connect thereto;

(b) receiving a connect message from the first partition that identifies a memory location of the shared memory region at which the first partition has allocated a first data area to serve as a buffer for transferring data from the first partition to the second partition;

(c) matching the received connect message to the listening socket created in step (a);

(d) allocating a second data area in the shared memory region to serve as a buffer for transferring data from the second partition to the first partition;

(e) mapping both the first and second data areas into a process space of the listening socket;

(f) initializing the second data area; and

(g) returning a connected indication to the first partition and informing the application on the second partition that the emulated socket connection has been established.

110. (previously presented) The method of claim 109, further comprising performing the following steps on the first partition:

(a') receiving the request from the first application to establish the socket connection with the second application.

(b') creating an emulated connecting socket;

(c') allocating the first data area in the shared memory region;

(d') sending the connect message to the second partition that identifies the memory location of the shared memory region at which the first data area has been allocated; and

(e') upon receipt of the connected indication from the second partition, mapping the first and second data areas into a process space of the connecting emulated socket to establish the emulated socket connection between the first and second partitions.

111. (currently amended) The method of claim 101, further comprising:

creating in the shared memory region, a plurality of output queues, one for each of said first second partitions, the output queue for a given partition indicating whether that partition has placed in the shared memory region a message intended for any of the other partitions and if so, identifying a buffer containing the message, each partition ~~poling~~polling the output queues of other partitions to determine whether those other partitions have placed any messages intended for it in the shared memory region;

receiving at the first partition from the first application, said request to send a message to the second application via the emulated network connection;

writing, in response to the received request, the message to an available buffer in the shared memory region and indicating in the output queue of the first partition that the message has been written thereto;

determining, at the second partition, from the output queue of the first partition, that the message has been placed in the available buffer and retrieving the message from the available buffer; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

112. (currently amended) A computer readable medium having program code store thereon for communicating between first and second applications executing on respective separate independent partitions of a partitionable computer system, wherein each of said first and second partitions operates under the control of a separate operating system, and wherein said first and second applications are configured to communicate with each other via a physical network using standard network interfaces, ~~without the need for providing an external network connection therebetween,~~ and wherein said program code, when executed, causes performance of the following:

receiving a request made by said first application for establishing a network connection with said second application for sending a message thereto; and

in response to said request, establishing an emulated network connection between said first and said second applications through a shared memory region of the computer system shared by both said first and second ~~partition~~partitions;

said establishing being such that said emulated network connection permits said first and second applications to communicate with each other using standard network interfaces even though there is no physical network therebetween;

said establishing also being such that the receiving of a message by said second application from said first application via said emulated network connection appears to said second application as having been sent via an external physical network even though there is no external physical network.

113. (previously presented) The method recited in claim 112, wherein said emulated network connection requested by said first application comprises a network socket connection, and wherein the step of establishing a connection through the shared memory region comprises establishing a connection through the shared memory region that emulates a network socket connection.

114. (previously presented) The method recited in claim 112, wherein said establishing communication between said first and second applications includes:

creating a data structure in said shared memory region comprising a plurality of data segments for use in sending a message from said first application to said second application.

115. (currently amended) The method recited in claim ~~103~~113, wherein sending a message from said first application to said second application includes:

writing, from the first partition on behalf of the first application, a message to one or more of said data segments, as needed, and updating an indication of the data segment containing the most recently written portion of the message;

reading from the second partition on behalf of the second application, a message from said one or more data segments and updating an indication of which data segments have been read from the data structure; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

116. (previously presented) The computer readable medium of claim 112, wherein said plurality of data segments form a circular buffer, and wherein updating an indication of the data segment containing the most recently written portion of the message comprises incrementing a head index.

117. (previously presented) The computer readable medium of claim 112, wherein updating an indication of which data segments have been read from the data structure comprises incrementing a tail index.

118. (previously presented) The computer readable medium of claim 117, wherein the program code, when executed, further causes the processor to poll, by the receiving partition, the shared memory region to determine if the message has been written to the shared memory region.

119. (previously presented) The computer readable medium of claim 118, wherein the program code, when executed, further causes the processor to receive, by the receiving partition, an interrupt initiated by the sending partition and indicating that the message has been written to the shared memory region.

120. (previously presented) The method of claim 113, wherein establishing an emulated socket connection between the first partition and the second partition further comprises performing the following steps on the second partition:

(a) creating an emulated socket connection on behalf of the second application to listen for attempts to connect thereto;

(b) receiving a connect message from the first partition that identifies a memory location of the shared memory region at which the first partition has allocated a first data area to serve as a buffer for transferring data from the first partition to the second partition;

(c) matching the received connect message to the listening socket created in step (a);

(d) allocating a second data area in the shared memory region to serve as a buffer for transferring data from the second partition to the first partition;

(e) mapping both the first and second data areas into a process space of the listening socket;

(f) initializing the second data area; and

(g) returning a connected indication to the first partition and informing the application on the second partition that the socket connection has been established.

121. (previously presented) The method of claim 120, wherein the step of establishing a connection further comprises performing the following steps on the first partition:

(a') receiving the request from the first application to establish the socket connection with the second application;

(b') creating a connection socket;

(c') allocating the first data area in the shared memory region;

(d') sending the connect message to the second partition that identifies the memory location of the shared memory region at which the first data area has been allocated; and

(e') upon receipt of the "connected" indication from the second partition, mapping the first and second data areas into a process space of the connecting socket to establish the socket connection between the first and second partitions.

122. (previously presented) The computer readable medium of claim 112, wherein the program code, when executed, further causes performance of the following:

creating, in the shared memory region, a plurality of output queues, one for each of said first and second partitions, the output queue for a given partition indicating whether that partition has placed in the shared memory region a message intended for any of the other partitions and if so, identifying a buffer containing the message, each partition polling the output queues of the other partitions to determine whether those other partitions have placed any messages intended for it in the shared memory region;

receiving at the first partition from the first application, said request to send a message to the second application via the requested type of network connection;

writing, in response to the received request, the message to an available buffer in the shared memory region and indicating in the output queue of the first partition that the message has been written thereto;

determining, at the second partition, from the output queue of the first partition, that the message has been placed in the buffer and retrieving the message from the buffer; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

123. (currently amended) A computer system comprising:

a plurality of processing modules, groups of one or more processing modules being configured so as to provide at least first and second separate independent partitions within the computer system, each of said first and second partitions operating under the control of a separate operating system, said first partition providing for executing a first application and said second partition providing for executing a second application, each of said first and second applications being configured to communicate with the other application via a physical network using standard network interfaces; ~~without the need for providing an external network connection therebetween~~

a main memory to which each processing module is connected, the main memory having defined therein at least one shared memory region to which at least said first and second ones of said partitions have shared access; and

program code, executing on each of at least said first partition and said second partition of the computer system, said program code establishing a connection between a said first application on said first partition and said second application on said second partition through the at least one shared memory region, wherein the connection through the shared memory region emulates a network connection requested by one of said applications, such that said first and second applications communicate with each other using standard network interfaces even though there is no physical network therebetween, and also such that the receiving of a message by said second application from said first application appears to said second application as having been sent via an external physical network even though there is no external physical network;

wherein said program code executing on each of said first and second partitions comprises a shared memory service provider that serves as an interface between a

component of the computer system that provides an API through which said first application can make said request for a network connection and the shared memory region of the main memory through which the emulated network connection is established;

wherein the shared memory service provider on each of said first and second partitions establishes a data structure in the shared memory region through which data is transferred from that partition to the shared memory service provider on the other partitions.

wherein the data structure comprises:

a plurality of data segments, each of the plurality of data segments for storing network message data to be sent from a sending shared memory service provider to a receiving shared memory service provider;

a control segment for controlling reading and writing of data in the plurality of data segments, the control segment comprising:

a first portion comprising:

a first field for storing an indication of the data segment containing the most recently written network message data; and

a second field for storing an indication of the data segment containing the earliest written, but not[[ot]] read, network message data;

and

a plurality of second portions, each second portion corresponding to one of the plurality of data segments for control of the data segment, each second portion comprising:

a first field for storing an indication of the beginning of network message data within the data segment; and

a second field for storing an indication of the end of network message data within the data segment.

124. (previously presented) The computer system recited in claim 123, wherein the first portion further comprises:

a third field for storing an indication that the sending shared memory service provider is waiting to send the network message; and

a fourth field for storing an indication that the receiving shared memory service provider is waiting to receive the network message.

125. (previously presented) The computer system recited in claim 123, wherein each second portion further comprises a third field for storing an indication of a length of network message data within the data segment.

126. (previously presented) The computer system recited in claim 125, wherein the plurality of data segments are linked to form a circular buffer, and wherein each second portion further comprises:

a fourth field for storing an indication of the next data segment in the circular buffer, and

a fifth field for storing an indication that the data segments contains a last portion of a network message stored across a plurality of data segments.

[[126]]127. (currently amended) The computer system recited in claim 123, wherein the computer system provides a resource through which the shared memory service provider can establish the data structure and control the transfer of data through it, the resource providing the ability to perform at least one of the following operations on the shared memory region: (i) allocate an area of the shared memory region; (ii) map and unmap an area of the shared memory region; deallocate an area of the shared memory region; (iii) send and receive signals to and from other partitions via the shared memory region; and (iv) receive status information about the shared memory region and about selected partitions.

[[127]]128. (currently amended) The computer system recited in claim 123, wherein the shared memory service provider comprises:

a dynamic link library (DLL) that executes in a user mode of the operating system of its respective partition, there being an instance of the shared memory service provider DLL in a process space of each application in the partition that may request the establishment of a network connection; and

a device driver that executes in a kernel mode of the operating system of the respective partition, there being only one instance of the device driver in each partition.

[[128]]129. (currently amended) The computer system recited in claim 123, wherein the connection established through the shared memory region emulates a network socket connection.

[[129[[130. (currently amended) The computer system recited in claim 128, wherein said program code executing on each of said at least first and second partitions comprises a shared memory service provider that serves as an interface between a component of the computer system that provides an API through which an application can make a request for a network socket connection and the shared memory region of the main memory through which the emulated network socket connection is established.

[[130]]131. (currently amended) The computer system recited in claim 129, wherein the operating system in each partition comprises a MICROSOFT WINDOWS operating system, and wherein the component of the computer system that provides the API of the requested socket connection comprises a Winsock DLL and a Winsock Switch, the Winsock DLL forwarding a request for a socket connection made by an application in a given partition to the Winsock Switch, which Winsock Switch allows multiple service providers, each of which provide TCP/IP services, to service such a request, and wherein the shared memory service provider acts as a TCP/IP service provider so that a request from an application for a socket connection can be serviced by the shared memory service provider.

[[131]]132. (currently amended) The computer system recited in claim 130, wherein the shared memory service provider on a first partition that represents the listening side of a requested socket connection performs the following steps:

(a) creating an emulated network socket on behalf of first application executing in the first partition in order to listen for attempts to connect thereto;

(b) receiving a connect message from the shared memory service provider on the second partition that identifies a memory location of the shared memory region at which the shared memory service provider on the second partition has allocated a first data area to serve as a buffer for transferring data from the second partition to the shared memory service provider on the first partition;

(c) matching the received connect message to the listening emulated network socket created in step (a);

(d) allocating a second data area in the shared memory region to serve as a buffer for transferring data from the first partition to the shared memory service provider on the second partition;

(e) mapping both the first and second data areas into a process space of the listening socket;

(f) initializing the second data area; and

(h) returning a "connected" indication to the shared memory service provider on the second partition and informing the application on the first partition that a socket connection has been established.

[[132]]133. (currently amended) The computer system recited in claim 131, wherein the shared memory service provider on the second partition performs the following steps:

(a') receiving a request from an application on the second partition to establish an emulated network socket connection with the first application on the first partition;

(b') creating an emulated network connection socket on the second partition;

(c') allocating the first data area in the shared memory region;

(d') sending the connect message to the first partition that identifies the memory location of the shared memory region at which the first data area has been allocated; and

(e') upon receipt of the "connected" indication from the first partition, mapping the first and second data areas into a process space of the connecting socket to establish the emulated network socket connection between the first and second partitions.

[[133]]134. (currently amended) The computer system recited in claim 123, wherein the program code implements a polling process by which each partition polls an area within the shared memory region to determine whether any communications intended for it have been placed in the shared memory region by another partition.

[[134]]135. (currently amended) The computer system recited in claim 133, wherein the area comprises a plurality of output queues, one for each partition, the output queue for a given partition indicating whether that partition has placed in the shared memory region any communications intended for any of the other partitions, each partition polling the

output queues of the other partitions to determine whether those other partitions have placed any communications intended for it in the shared memory region.

[[135]]136. (currently amended) The computer system recited in claim 134, wherein for any communications placed in the shared memory region by a sending partition and intended to be received by another partition, the output queue of the sending partition specifies the location within the shared memory region of a buffer containing that communication.

[[136]]137. (currently amended) The computer system recited in claim 135, wherein the program code executing on each of said first and second partitions further comprises a shared memory driver that receives a request to send a message to an application on another partition, the request having been made in accordance with the application programming interface (API) associated with the requested type of network connection, and that, in response to the request, causes the message to be placed in an available buffer in the shared memory region and causes an indication of the message to be placed in the output queue of the sending partition.

[[137]]138. (currently amended) The computer system recited in claim 136, wherein the shared memory driver on each partition implements a same interface as a network device driver to enable application programs and the operating system on that partition to send communications to other partitions via the shared memory region in the same manner that communications are sent to other computer systems over a network via a network interface card.